DOCUMENT RESUME

ED 281 486 IR 012 602

AUTHOR Cleaver, Thomas G.
TITLE Simulated Laboratory in Digital Logic.
INSTITUTION Louisville Univ., KY. Dept. of Electrical Engineering.
PUB DATE 85
NOTE 85p.
AVAILABLE FROM Thomas G. Cleaver, Department of Electrical Engineering, University of Louisville, Louisville, KY 40292 ($7.00).
PUB TYPE Guides - Classroom Use - Materials (For Learner) (051) -- Reports - Descriptive (141)

EDRS PRICE MF01/PC04 Plus Postage.
DESCRIPTORS *Computer Science Education; *Computer Simulation; Computer Software; Course Descriptions; *Electric Circuits; *Engineering Education; Higher Education; Laboratory Manuals; *Learning Laboratories; Microcomputers
IDENTIFIERS *Computer Assisted Design; Electrical Engineering; Robot Odyssey I

ABSTRACT

Design of computer circuits used to be a pencil and paper task followed by laboratory tests, but logic circuit design can now be done in half the time as the engineer accesses a program which simulates the behavior of real digital circuits, and does all the wiring and testing on his computer screen. A simulated laboratory in digital logic has been proposed at the University of Louisville (Kentucky) in order to provide students, who presently do not have access to such software tools and workstations because of their expense and complexity, with a simplified and easy-to-learn digital logic design program that runs on an inexpensive microcomputer. This laboratory will provide students with experience with modern techniques and equipment, the opportunity to design the complex circuitry this equipment makes possible, and the opportunity to learn how to deal with the timing problems so common in complex digital circuits. The logic simulator program, Robot Odyssey I, has been acquired and a laboratory manual written. Evaluation of the laboratory and design of suitable measurements of student mastery of behavioral objectives are still needed. The course proposal and syllabus serve as an introduction to the laboratory manual, which makes up the major part of this report. (MES)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
X This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED281486

SIMULATED LABORATORY IN DIGITAL LOGIC

ABSTRACT

1985

The design of computer circuits used to be a pencil and paper task followed by laboratory tests. The modern engineer has found a better way. Logic circuit design and test can now be done by simulation at a computer "workstation". The engineer accesses his "software tool" (a program which simulates the behavior of real digital circuits), and does all the wiring and testing on his computer screen. His design is completed in half the time it would have taken using the old methods.

Students do not have access to the software tools and the workstations because they are too expensive and difficult to teach. As a result, graduating students have no experience with the modern techniques and equipment, have not designed the kinds of complex circuitry this equipment would make possible, and have not learned how to deal with the timing problems so common in complex digital circuits.

This proposal intends to address these problems by providing students with a simplified and easy-to-learn digital logic design program which runs on an inexpensive microcomputer. Learning would take place in a Simulated Laboratory in Digital Logic. A simplified logic simulator program has already been acquired, and a lab manual has been written. What remains is the evaluation of the laboratory and the design of suitable measurements of student mastery of the behavioral objectives.

IR012602

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Thomas G. Cleaver
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

SIMULATED LABORATORY IN DIGITAL LOGIC

The Logic Design Course

Approximately 200 electrical engineering and computer science sophomores and juniors enroll in EE 310, Logic Design, each year. The course begins with computer math and goes on to the design of digital logic circuits, which are the building blocks of computers. The mathematical, electrical, and timing behaviors of simple digital logic circuits are investigated. The course concludes with the design of more complex digital circuits. This material provides the underpinnings for the students' future study in computer design.

Logic Design is taught as a three semester hour lecture course, with the usual quizzes, tests, and homework, as described in the attached syllabus. A separate course, EE 311, Logic Design Laboratory, is a required corequisite of Logic Design. It gives students laboratory experience in digital logic circuit design and construction.

The goals of the Logic Design course are to teach the student to:

* do mathematics in the binary, octal, and hexadecimal number systems;
* perform digital logic operations using Boolean algebra;
* design and implement simple digital logic circuits;
* understand and consider the importance of timing in the the design of such circuits;
* extend the above to the design and implementation of more complex digital circuits; and
* use computer-aided engineering (CAE) logic design tools to design and test logic circuits.

The last three goals have not been successfully met in the past, and it is to these goals that this proposal is addressed.

Teaching/Learning Problems in Logic Design

The world of the engineering professional is becoming more and more complex. This is especially true in electrical engineering and computer science, in which the professional routinely encounters "chips" implementing up to a million transistors. To assist in dealing with this complexity, the tools of computer-aided engineering (CAE) have been developed. CAE tools frequently take the form of software packages which allow the engineer to design circuit boards, integrated circuit chips, and mechanical devices at a computer terminal rather than in the laboratory. Typically, the engineer will sit down at his "work-station" and manipulate symbols representing transistors, logic gates, or mechanical interconnections to create a model of an electronic circuit or mechanical device. He will then run

IR012602

simulated tests of his design, and finally, he will produce the specifications and documentation necessary to produce the "real" product or device.

It is our desire to train electrical engineers and computer scientists who can function in this sophisticated environment. Toward this end, the Speed Scientific School is planning to build a Computer-Aided Engineering Center, and the faculty are encouraged to provide CAE experiences for their students. There are many obstacles to providing this experience in this Logic Design class:

* CAE workstations for logic circuit design are usually special purchase items costing from $5000 to $75,000 each.
* Logic design software is expensive - $2000 to $20,000.
* It takes a long time to learn how to use most CAE software packages, far longer than the time that could be allotted from a logic design class.

The lack of this CAE experience in logic design means that, in addition to the lack of exposure to modern CAE tools, the students will not get experience in designing complex circuits (the hardware lab only allows time for the design of simple circuits), the "chip" concept (putting several logic devices within a single integrated circuit) is poorly understood, and the intricacies of the timing of pulses within circuits are not understood at all.

Objectives of the Proposal

This proposal seeks to address the above deficiencies in a cost-effective way by providing the students with a much-simplified CAE tool that is inexpensive, runs on existing microcomputers, and is easy to learn. Furthermore, the CAE tool can be integrated into the Logic Design course by providing a "Simulated Laboratory" experience through experiments run on the microcomputers.

It should be understood that the proposed simulated laboratory would be a part of EE 310, Logic Design, and is not intended as a surrogate for EE 311, Logic Design Lab. The latter is a hardware lab in which simple digital logic circuits are actually constructed from integrated circuits. EE 311 has different goals than EE 310, but they are equally important.

Methodology

The plan for the Simulated Laboratory in Digital Logic calls for the selection or design of an appropriate CAE logic design tool and the preparation of a set of experiments for the lab. Fortunately, a suitable commercially available software tool was found after study of several products. Robot Odyssey I, which runs on the Apple II series microcomputers, is touted as "a revolutionary robot construction set and adventure game." It is

all that and more. The front side of the disk contains an adventure game in which digital logic circuits are constructed and installed in robots. Although this "game" is highly educational, it will not be used. The back side of the disk is of greater interest. It contains three tutorials which familiarize the user with the CAE tool and digital circuit design, and an "Innovation Lab" in which logic circuits can be built, tested, and installed in simulated integrated circuits.

The Robot Odyssey software is truly a CAE logic design tool, albeit a much simplified one. The students can indeed design and construct simulations of even rather complex logic circuits. Figure 1 shows an example of such a complex circuit created with Robot Odyssey. Note that the circuit contains many components and interconnections. The entire circuit can be "shrunk" and treated as a single "chip". The students can test circuit function by applying electrical signals to the inputs and observing the electricity flow within a circuit. This capability of monitoring the electricity flow in a circuit means that circuit timing can be directly observed and measured; this is impossible in the classroom or hardware lab.

Students in Logic Design will use a laboratory manual and the Robot Odyssey software to perform a series of experiments throughout each term. To perform an experiment, the student will go to one of two computer centers on campus, check out the Robot Odyssey disk, boot the disk on an Apple II microcomputer, and carry out the instructions in the laboratory manual.

Formative Evaluation

Testing of the ideas presented in this proposal has already begun. Six copies of the Robot Odyssey software have been obtained and a lab manual has been written, a copy of which is attached to this proposal. In the fall of 1985 and the spring of 1986 students in Logic Design participated in the simulated lab. Each laboratory experiment included a "critique" section which encouraged the students to make suggestions for improving the laboratory experience. Analysis of their work and the critiques has resulted in updates and corrections to the lab manual. This amounts to "field evaluation" of the instructional methodology.

Evaluation of Success of the Simulated Laboratory

Although much progress has been made in the design and implementation of the simulated lab, more work needs to be done to determine if it is suitable for meeting the goals set forth above. In order to determine if the goals are met, it will be necessary to measure the skills of the students upon completion of the instruction. The goals above are easily translatable into specific performance objectives; after completing the simulated laboratory component of the Logic Design course, the student should be able to:

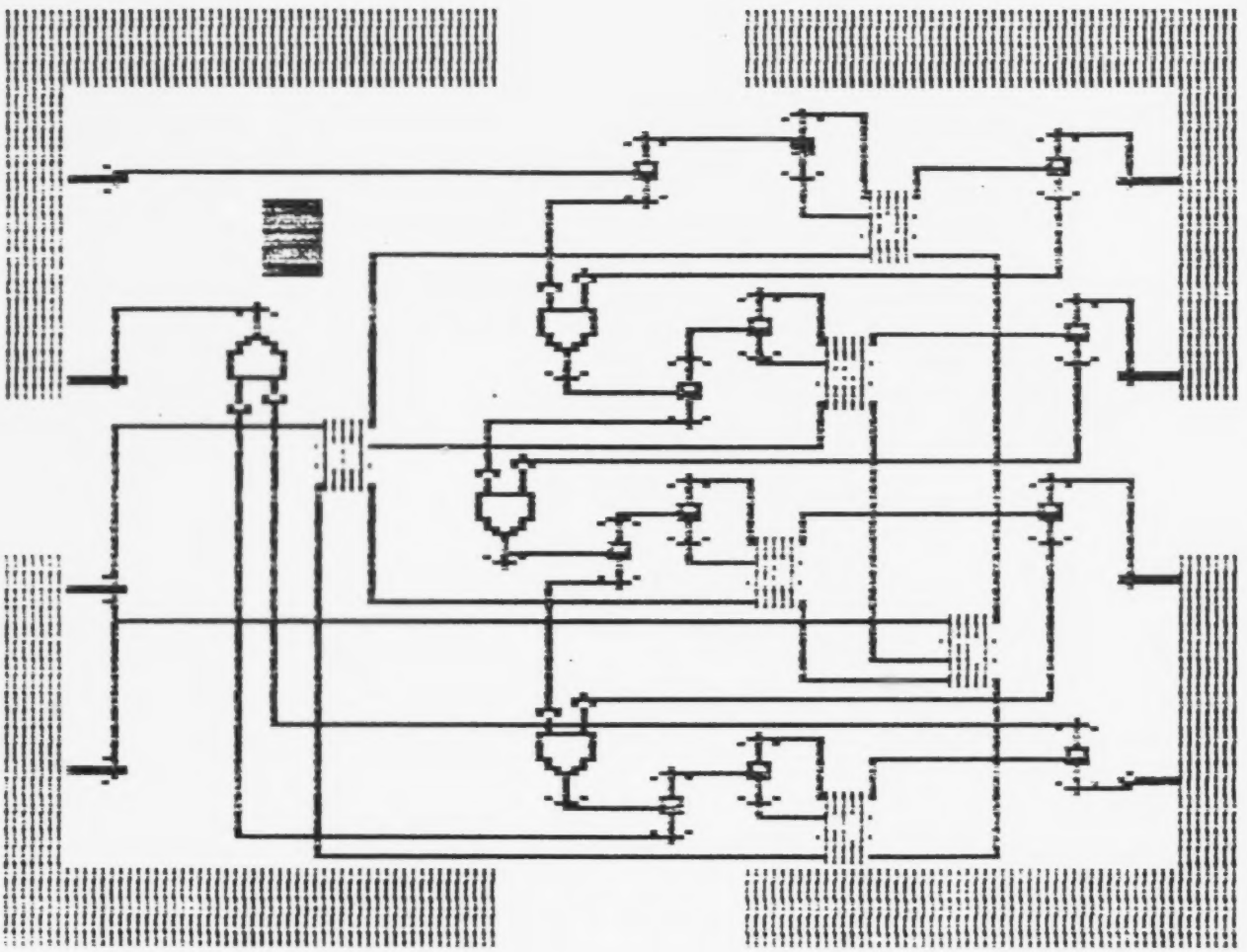


Figure 1. Synchronous 4-Bit Binary Counter Created with Robot Odyssey I

* design logic circuits which are free of timing problems;
* detect timing errors in existing logic circuits;
* design complex logic circuits such as multiplexers using a CAE logic design tool; and
* test existing logic circuits for proper function using a CAE logic design tool.

In order to measure the performance attained by the students, it will be necessary to develop the appropriate measurement tools. These tools will probably include lab reports and objective tests.

It is believed that the existing simulated laboratory experiments successfully teach the students to do these things at a satisfactory level, but their success at these tasks is not specifically measured, nor are the experiments optimized to teach these concepts. Therefore, the experiments will be modified and a criterion-referenced testing scheme will be developed.

Proving that the appropriate learning has taken place may be difficult, but one factor that makes it somewhat easier is that it can be assumed with a fair degree of certainty that upon entry to the Logic Design course, the student has no skills in the appropriate areas. Logic Design is an entry-level course, and with the exception of the few students who retake this course and the few students who have studied computer circuits as a hobby, the students have no skills in logic circuit design.

The assessment of the level of mastery of the required skills is the biggest problem. The lab reports submitted by the students will certainly be useful in this assessment. Criteria-referenced test items will also prove useful.

As the author is not an expert in the theory of instructional design, he will seek aid in the carrying out of the objectives of this proposal, both from reference works and from acknowledged experts within the Metroversity.

Required Technology

The Robot Odyssey program requires Apple II series computers with at least 64 kilobytes of memory for student use.

Cost of Implementation

Assuming the Apple computers are available (at least one for every 10 students in the class), which is the case on most modern campuses, the costs are as follows:

Six copies of Robot Odyssey I software.....$109
Simulated Lab workbooks.....$3.00/student.

ATTACHMENT I - SYLLABUS

EE 310 LOGIC DESIGN SPRING 1986

Meets MWF 8:00 - 8:50 AM in W. S. Speed 010

Instructor: T. G. Cleaver: Room 203 W. S. Speed,

Texts:
1. Mano: Digital Logic and Computer Design (Prentice-Hall, 1979)
2. The TTL Data Book, Vol. 2, Standard TTL, Schottky, Low-Power Schottky Circuits (Texas Instruments, 1985)

Welcome to Logic Design! Please read this handout carefully and ask about anything that is not clear.

The course has been structured to facilitate your learning by providing a variety of design and analysis experiences involving paper-and-pencil problem solving, and simulated laboratory experiments on a computer. Lectures will be devoted to clarifying the text material, to providing additional material (through lectures or handouts), to answering questions from the class, and to working example problems. Reading and assignments should be completed prior to the class meeting for which they are assigned so that you can derive the greatest benefit from the lectures. Homework problems will complement the reading assignments and should be attempted along with the reading.

TOPICS: We will cover topics from both Combinational and Sequential Logic.

Combinational Logic: Number Representations, Codes, Boolean Algebra and Elementary Gates, SSI Realizations of Gates, Logic Circuits, Simplification of Boolean Functions, Some MSI and LSI Functions; Applications. Chapters 1-5 of Mano.

Sequential Logic: Latches, Flip-Flops, Counters, Shift Registers, Memories, Timing Signal Generators, State Machines; Applications. Chapters 6,7 of Mano.

HOMEWORK: Paper-and-pencil problems will be assigned and collected. They will count for 20% of your grade. You are expected to do these problems on your own. Solutions to the homework problems will be posted on my bulletin board. These solutions are NOT TO BE REMOVED from the bulletin board, even temporarily, FOR ANY REASON. Students who maintain an 80% or better quiz average need not turn in the homework, but will be given full credit for it. Whatever your quiz average, you should work the assigned problems since they will aid in your learning and since quiz problems may closely resemble homework problems. All students must turn in all the homework prior to the first quiz.

EXAMS: Seven 30-minute quizzes will be given. These quizzes will count for 40% of your grade. In computing your total points, the score on the lowest quiz will be replaced by the average of the scores on your highest and lowest quizzes.

Assume that all quizzes will be closed book, unless announced otherwise. Some of the quiz problems may involve only minor changes upon the homework problems. Solutions to the quizzes will be posted outside the classroom and on my bulletin board following the quizzes. Do not remove these solutions for any reason.

Make-up quizzes will be given only if previously arranged with the instructor or for certain documentable extraordinary circumstances. Make-ups are often more difficult than the original quizzes.

FINAL EXAM: A final exam will be given; all students are required to take it. The final will count for 20% of your grade.

SIMULATED LABORATORY: A set of simulated laboratory experiments has been prepared for use on the Apple II computer. Each student will be given a printed set of the experiments and a schedule for their completion. When you wish to run an experiment, go to the Apple Orchard or the South Center and sign out the Robot Odyssey I software from the monitor on duty. Leave your ID card as collateral. The term will conclude with a design contest based on the components of the simulated laboratory environment. The simulated laboratory will count for 20% of your grade. You are expected to do these labs on your own.

GRADING SUMMARY: Course grades will be assigned according to the chart below. Grades will not be curved.

Percent	Grade
90-100	A
80- 89.9	B
70- 79.9	C
60- 69.9	D
Below 60	F

PICKING UP GRADED ASSIGNMENTS: Graded homework, quizzes, and simulated labs may be picked up outside my office. In most cases, assignments will be graded within 2 days of when they are turned in. NEVER pick up another's assignment without his permission.

PUNCTUALITY: You are expected to be on time for class. Assignments are due BEFORE the beginning of the class. Assignments will not be accepted after class begins.

PENALTIES: A student having a non-excused exam absence will not be permitted a make-up. Students caught cheating on exams,submitting copied assignments, or otherwise engaging in unethical behavior will receive a failing grade for the course.

TGC/sbt

ATTACHMENT II

Simulated Laboratory In Digital Logic

Thomas G. Cleaver

UNIVERSITY of LOUISVILLE

Simulated Laboratory in Digital Logic

by Thomas G. Cleaver

A set of experiments in digital logic design based on the Innovation Laboratory program in Robot Odyssey I.

Robot Odyssey I is a product of The Learning Company.

© 1985, Thomas G. Cleaver

THE UNIVERSITY OF LOUISVILLE
Department of Electrical Engineering
Louisville, Kentucky 40292

PREFACE

The experiments contained herein are intended to teach the basic concepts of digital logic design. Students will learn through experiment about truth tables, logic gates, and flip-flops. Although the textual material provides sufficient information for performing the experiments, this book is intended to be used in conjunction with the first lecture course in digital logic as taught to engineers, computer scientists, and data processing technicians.

"Simulated" experiments, as used here, are those which are performed not in a real laboratory using real chips, but in a computer simulated laboratory using chip models. The Innovation Laboratory contained in the Robot Odyssey I software provides the simulated laboratory environment. Students can work with AND, OR, NOT, and XOR gates and can use these gates to "wire" together circuits of great complexity using a simulated soldering pen. Although the value of actual hands-on laboratories cannot be overemphasized, the Innovation Lab provides certain distinct advantages over physical labs: wiring and testing of circuits can be done much faster; students can "see" the logic levels on each wire in the circuit (logic 0 is white, logic 1 is orange); the propagation delay of the simulated gates is about 0.3 seconds, far longer than the delays of real devices, which makes it possible to observe the sequence of events in a logic circuit and to design precise timing circuits; and complex circuits can be designed and "burned" into chips for later use, and even nested inside other chips.

Use of these experiments requires a 64K Apple II computer, a disk drive, and the Robot Odyssey I software. A color monitor is highly desirable. A mouse or joystick is also desirable. Each user should have one or more blank disks for storage of chip designs.

Each of the experiments can be completed in 1 to 2 hours.

CONTENTS

		Page
Preface		iii
Table of Contents		v
Command Summary		vii
Experiment 1.	Software and Toolkit Familiarization	1-1
Experiment 2.	Basic Gates	2-1
Experiment 3.	Fundamental Boolean Operations	3-1
Experiment 4.	Chip Design	4-1
Experiment 5.	Binary Adders	5-1
Experiment 6.	Two-Digit Binary Adder	6-1
Experiment 7.	Decoders and Demultiplexers	7-1
Experiment 8.	RS Flip-Flop	8-1
Experiment 9.	Gated and Clocked Flip-Flops	9-1
Experiment 10.	D and JK Flip-Flops	10-1
Experiment 11.	Registers	11-1
Experiment 12.	Counters	12-1
Experiment 13.	Memory	13-1
Experiment 14.	Robot Control	14-1

COMMAND SUMMARY

space bar	Pick up/drop, on/off, etc.
I, J, K, M	Movement
CTRL-I, J, K, M	Movement in small steps
C	Use cursor
H	Toggle hot cursor
L	Load a chip
P	Use paint brush
R	Use remote control
S	Use solder pen or save chip
T	Summon or open tool kit
?	Access chip records/help
ESC	Return to menu/save a lab
CTRL-S	Toggle sound effects

EXPERIMENT 1

SOFTWARE AND TOOLKIT FAMILIARIZATION

NAME_____ DATE_____

OBJECTIVES

To familiarize you with the Robot Odyssey I software.
To provide you with practice in using the keyboard commands.
To familiarize you with basic logic gates.
To give you practice in wiring simple logic circuits.

PREPARATION

Read pp. 5-8 and 29-43 in the Robot Odyssey I manual.

PROCEDURE

In this experiment and the others that follow you are encouraged to go beyond what is required and "explore" on your own.

Boot Side 2 of the Robot Odyssey I disk, and select the Robot Anatomy tutorial. Notice that you can move more quickly through the "rooms" in the tutorial by holding down the I, J, K, M keys (Apple IIe) or using the REPT key (Apple II+). Press CTRL-S to quiet the lesson if the program's sound effects are bothering those around you.

You will be able to observe the logic state of any wire, gate, etc. by its color. Logic 0 shows as white and logic 1 as orange. If you are using a monochrome monitor, logic 1 will appear as thinner dashed lines. The tutorial speaks of "electricity flow" as orange. This is the tutorial's way of defining a logic 1. All of the robot's parts are active in the logic 1 state. In the real world, some devices are activated by logic 0.

Does the robot's antenna have an input or an output?

Note that inputs are always wired to outputs. An input can never be wired to another input, nor an output to another output. It is therefore impossible to wire short circuits, unlike the real world.

The wall-following robot has a single logic gate inside. What kind is it?

Although the specifics of robot construction discussed in this tutorial may not be pertinent to logic design, you will need the techniques studied here for use in future experiments.

When finished, exit from the Robot Anatomy tutorial and select the Toolkit tutorial.

The first circuit you encounter is an inverter (NOT gate) connected so that its output feeds its own input. This causes the circuit to oscillate between the 1 and 0 states at a rate of about 3 cycles per second. A real device connected in this way will also oscillate, but the frequency of oscillation will be much faster: about 50,000,000 cycles per second.

"Freezing" the electricity, as described in the tutorial, is not possible in the real world. This capability will allow us to examine the logic levels in a circuit at any instant in time. Remember that orange represents a logic 1 and white a logic 0.

When using the solderpen with a monochrome monitor, it may be difficult to tell when the tip "glows orange". In monochrome this can be observed as a slight rightward shift of the tip. Practice with the solderpen until you can make it operate reliably.

The "hot" cursor applied to an input is the real-world equivalent of applying a logic 1 (usually +5V).

What effect does the hot cursor have if the remote control is off?

When you come to the CROSSROADS, first study nodes, then logic gates, then flip-flops.

Nodes are used in much the same way as terminal strips. They allow one point to be connected to many others. After you have wired the key sensor to the input of the node, try wiring it to one of the outputs of the node instead. What happens?

When studying the logic gates, begin with the NOT gate, then go on to the AND gate, the OR gate, and finally, the XOR gate.

We use a prime (') to indicate the NOT operation when writing logic expressions. F = A' means that F (the output of a NOT gate) is the inverse of A (the input of the NOT gate). Another name for the NOT gate is the inverter. If the input to a NOT gate is a logic 1 (orange), what is the output?

The AND operation is indicated by a dot or asterisk in logic expressions. If an AND gate has two inputs, A and B, and one output F, the logic expression is F = A * B or simply F = AB. If a logic 1 is provided to one input of an AND gate and a logic 0 is provided to the other, what will the logic level of the output be?

The operator for the OR gate is written as a plus (+) sign. If an OR gate has A and B as its inputs and F as its output, its logic expression is F = A + B. If a logic 1 is provided to one input of an OR gate and logic 0 is provided to the other, what will the logic level of the output be?

Although the AND and OR logic expressions look like algebraic multiplication and addition, they are not the same and should not be confused.

The AND, OR, and NOT gates are usually considered to be the most fundamental logic gates. The XOR (exclusive OR) is less common, but it is still a very useful logic element. The operation for the XOR operation is written as a plus sign within a circle. If an XOR gate has inputs A and B and output F, its logic expression is F = A $\oplus$ B. This means "A or B, but not both". An equivalent logic expression is F = A'B + AB'. Here is another equivalent expression: F = (A + B)(AB)'.

If one input of an XOR is a logic 1 and the other is a logic 0, what will the output be?

Flip-flops will be studied in more detail later in the experiments on sequential circuits. For now, a general familiarity is sufficient.

Now that you have finished the tutorial, try exploring on your own. Discuss your findings below.

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 2

BASIC GATES

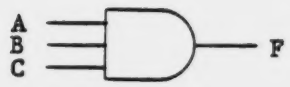
NAME_____ DATE_____

OBJECTIVES

To introduce the truth table concept.
To investigate the functions of the basic gates.
To investigate the behavior of simple combinational circuits.
To learn the function of NAND and NOR gates.
To verify their operation with truth tables.

PREPARATION

One way of defining the behavior of a logic circuit is with a truth table. The truth table is a list of all possible circuit inputs with their corresponding outputs. Shown below is the truth table for a 3-input AND gate.



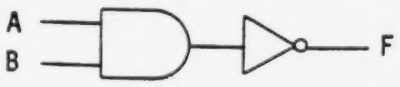
Inputs			Output
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Three-input AND Gate and its Truth Table

Note that all three inputs must be at the logical 1 level before the output becomes a logic 1. Also note that all 8 possible input combinations are shown. A gate with n inputs will have 2^n entries in its truth table.

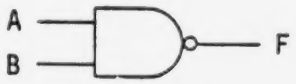
The logic expression for this AND gate is F = A*B*C, or simply F = ABC.

Many real-world logic chips are based not on AND and OR, as we have stuied so far, but on NAND and NOR. The NAND gate is a combination of an AND gate with an inverter, as shown below.



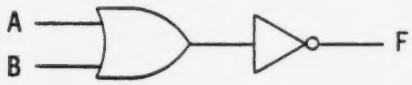
Two-input NAND Gate Construction

It is usually drawn this way: .



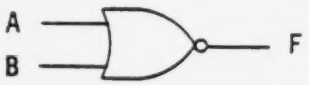
Two-input NAND Gate Circuit Symbol

Similarly the NOR gate is a combination of an OR gate with an inverter:



Two-input NOR Gate Construction

It is usually drawn this way:



Two-input NOR Gate Circuit Symbol

The truth tables for these two gates are shown below along with their logic expressions.

NAND
F = (AB)'

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR
F = (A + B)'

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

NAND and NOR Gate Truth Tables

Note that the output of the NAND gate is exactly the opposite the output of an AND gate. Likewise, the NOR gate's output is the opposite of the output of an OR gate.

PROCEDURE

Unlike the first experiment, this one does not use the tutorials on the disk.

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab. Tour the lab. Compare what you find with what is in the manual.

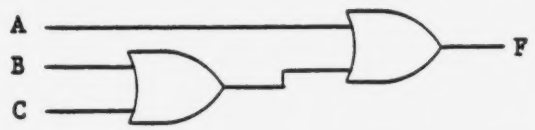
Take an AND gate from the tool kit and investigate its function with the hot cursor. Be sure the remote control is on.

Note that when nothing is connected to a gate input, the input will be a logic 0. This is usually not the case for real-world logic circuits. A "floating" input in such circuits usually presents a logic 1; to get a logic 0, the input must be "tied low", i.e., connected to ground.

Complete the truth table for the AND gate in the space below. Also write the logic expression for this gate.

Inputs		Output
A	B	F

Use the solder pen to wire the circuit shown below.



Test the circuit's function. To test all 8 possible input combinations you may need to use extra components. Toolkit flip-flops are useful for this purpose. You can use a flip-flop to generate either a 1 or a 0 to connect to any of the inputs (A, B, or C) as needed.

Based on your test, fill in the truth table below, and write the logic expression.

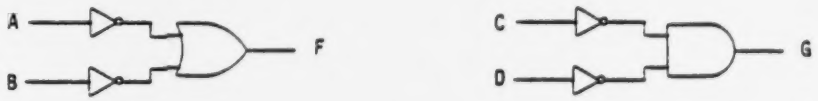
Inputs A B C	Output F

Use the solder pen to wire a circuit that will implement the 3-input AND function F = ABC. Sketch the circuit below. Test it to verify that it has the same truth table as given in the Preparation section of this experiment.

Take an AND gate and an inverter from the tool kit and wire them to form a NAND gate. Use the hot cursor to verify the truth table given in the Preparation section of this experiment. Enter your truth table below.

Take an OR gate and an inverter from the tool kit and wire them as a NOR gate. Use the hot cursor to verify the truth table given previously. Show your truth table below.

Wire the following two circuits:



Test these circuits with the hot cursor and complete their truth tables below.

A	B	F

C	D	G

What functions do these circuits perform? Discuss in comparison to the NAND and NOR circuits constructed earlier.

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 3

FUNDAMENTAL BOOLEAN OPERATIONS

NAME_____ DATE_____

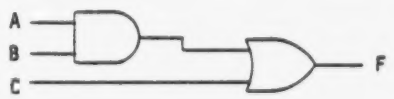
OBJECTIVES

To investigate Boolean expressions experimentally.
To implement simple Boolean expressions with logic gates.

PREPARATION

AND, OR, and NOT are the three basic operations of Boolean algebra. You have already been introduced to these operations individually, but now they will be combined to produce more complex Boolean logic expressions.

The circuit shown below shows an AND gate feeding an OR gate.



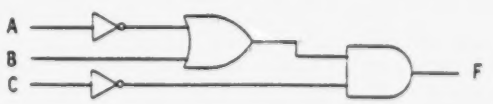
First A is ANDed with B, then the result is ORed with C. The logic expression for this is F = (A*B) + C, or simply F = AB + C.

By examining the circuit or the logic expression, we can determine the output for any given input. For example, if A = 0, B = 1, and C = 0, it follows that the output of the AND gate is 0, the two inputs to the OR gate are 0, and the output F is therefore 0. By going through all 8 possible input combinations, we could generate a truth table for this circuit.

In examining these circuits and expressions, you may wish to keep the following in mind:

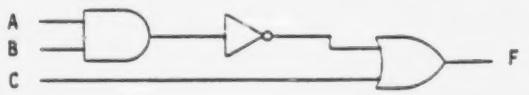
* If any input to an AND gate is 0, the output is 0.
* If any input to an OR gate is 1, the output is 1.
* The NOT gate just inverts (complements) its input.

More complicated circuits and expressions can be constructed by interconnection of logic gates, as shown in the example below.



The logic expression for this circuit is F = (A' + B)C'.

If you are given a logic expression, you can reverse the process and immediately draw the logic circuit. For example, the expression F = (AB)' + C can be drawn as shown below.

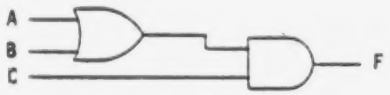


In the experiments below, you will construct several combinational logic circuits and test their operation.

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab.

Construct the circuit below from tool kit parts.



Before testing it with the hot cursor, analyze the circuit "by hand" and complete the truth table below.

A B C	F

Now test the circuit with the hot cursor to verify all 8 possible input combinations. Be sure the remote control is on. Does it agree with your truth table?

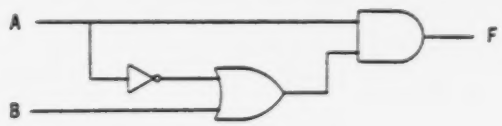
Complete the truth table below for the logic expression F = A'B + C'.

A	B	C	F

Construct the circuit which corresponds to this logic expression. Sketch it below.

Test your circuit with the hot cursor. Does it agree with your truth table?

Construct the circuit shown below. You will need a node (as well as gates) from the tool kit in order to do this.



Write the logic expression for this circuit.

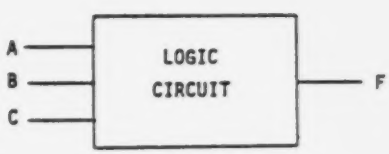
Complete the truth table below that would result from this circuit.

A	B	F

Test the circuit with the hot cursor. Does the test verify your truth table?

Compare the results from this last circuit with the results you would expect from a simple AND gate. Discuss.

Using toolkit parts, design and construct a logic circuit that will give a logic 1 output if at least one of its three inputs is a logic 0.



Although you may use any design you wish, the simplest design requires but three toolkit parts. Sketch (or provide a printout of) your circuit below, including labels and explanatory notes where necessary.

Test your circuit to be sure it functions as required.

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 4

CHIP DESIGN

NAME_____ DATE_____

OBJECTIVES

To introduce the chip design features of the Innovation Lab.
To give you practice in designing and "burning" chips.

PREPARATION

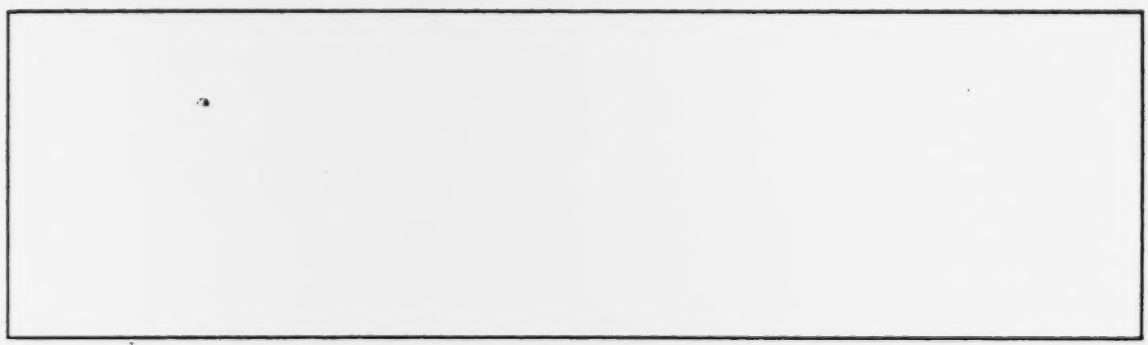
Read pp. 19, 21, and 22 of the Robot Odyssey I manual. Reread pp. 39-41.

You will need a blank diskette to do this experiment.

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Chip Design tutorial.

Carefully note the numbering scheme of the prototype chip. Sketch it below.



Real-world chips frequently have many more pins then the prototype chip, but the numbering scheme is similar. Fourteen- and sixteen-pin chips are quite common. Unlike the prototype chip, real-world chips must have pins for power (usually +5v) and ground.

When you solder, notice that the first point you solder produces a horizontal wire, and the next produces a vertical wire. You can use this information to make neater circuits.

After you have wired the small chip and the NOT gate to the prototype chip, sketch what the prototype chip looks like (from the outside).

Complete the tutorial (including the Paint Brush and Shape Editor) then go to the Innovation Lab.

Go to the Burner Room and pick up Chip 1. Press "?". What do you see?

Press "esc".

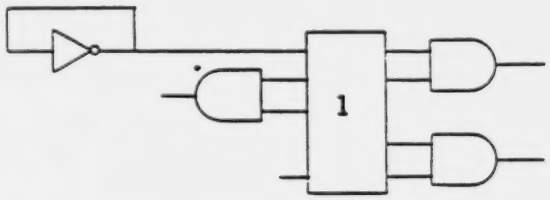
Press "L" and follow the directions. List the chips that are on side 2 of the Robot Odyssey I disk:

List the chips that are on side 1 of the Robot Odyssey I disk:

You may use some of these chips in future experiments.

Load the Bus Chip. Press "?". What do you see?

Wire this chip as shown below.



Be sure the remote control is on. Describe the results.

Pick up Chip 1. Press "S". Press "esc". Insert a blank disk into the drive (you will use this disk to save your chips and partially completed labs). Press "C". Press "Return". You have now created a storage disk. Save the Bus Chip on this disk (just for practice). Save the lab on this disk (also just for practice).

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 5

BINARY ADDERS

NAME_____ DATE_____

OBJECTIVES

To investigate binary addition.
To design circuits which accomplish binary addition.

PREPARATION

Arithmetic in binary is much the same as it is in decimal, but the inherent simplicity of the binary number system makes circuit implementation of binary operations much simpler than their decimal counterparts. In this experiment we will investigate binary addition and circuits which accomplish binary addition.

When adding two single-digit binary numbers, there are four combinations to consider:

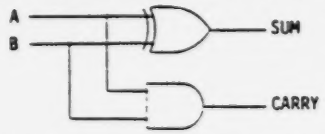
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

In the last of these four cases, we can see that a CARRY is generated as a result of the addition.

To implement this addition with a logic circuit, it is best to begin with a truth table.

A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The CARRY result can be seen to be nothing more than an AND of A and B, while the SUM is the XOR of A and B. We can therefore implement this addition with two gates as shown below.



Half-Adder Circuit

This is called a half-adder.

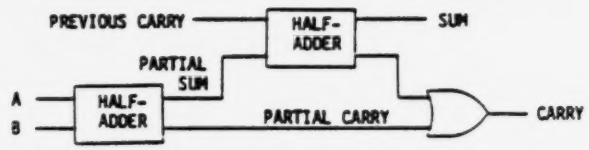
Now let us investigate multiple-digit binary addition. Consider this addition:

```
  1 0 1
+ 1 0 1
  -----
1 0 1 0
```

The addition in the first column generates a carry which must be added to the sum of the digits in the second column. In general then, the addition of two binary numbers requires the addition of successive columns includng the carry from the previous column. To accomplish the addition of each column we use a full-adder, which has the following truth table.

A	B	PREVIOUS CARRY	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

This truth table could be analyzed to develop logic expressions for the CARRY and SUM; however, there is an easier method. Consider the summation as two steps: first, adding A and B, then adding the PREVIOUS CARRY to the result. This can be accomplished with the logic circuit shown below.



Full-Adder Circuit

Spend a few minutes thinking about this logic circuit and considering how it works. Be sure you understand it before going on.

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab. Go inside the Prototype Chip.

Construct a half-adder with an XOR gate, an AND gate, and nodes from the toolkit. Use the hot cursor to complete its truth table below.

A	B	CARRY	SUM

Does this agree with the truth table in the Preparation section of this experiment?

Construct a second half-adder inside the Prototype Chip and connect the two half-adders and an OR gate to form a full-adder. Use the hot cursor to complete its truth table below.

A	B	PREVIOUS CARRY	CARRY	SUM

Does this agree with the truth table given in the Preparation section of this experiment?

Wire A to pin 1 of the Prototype Chip, B to pin 2, PREVIOUS CARRY to pin 3, CARRY to pin 7, and SUM to pin 8.

Go outside the Prototype Chip and verify that the Prototype Chip now functions in the same way as the full-adder circuit.

Go to the Burner Room and burn the Prototype Chip into a small chip. Pick up the small chip, press "?" and enter appropriate documentation for the chip. Copy below what you have entered for chip documentation (or provide a printout).

Verify that the small chip now behaves the same way as the full-adder circuit.

Save the chip on your Storage Disk. You will need it for future experiments.

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 6

TWO-DIGIT BINARY ADDER

NAME_____ DATE_____

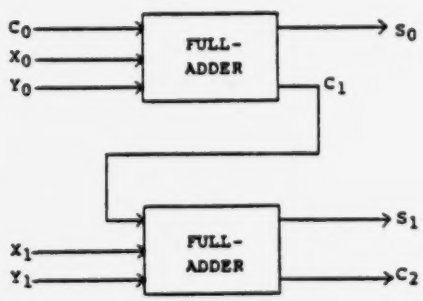
OBJECTIVE

To design a sophisticated adder chip.
To investigate the features of ripple carry.

PREPARATION

To perform any significant arithmetic, an adder should be able to handle more than two single-digit binary numbers. In this experiment you will design a chip that can add two two-digit numbers. You will also learn how to connect several of these chips together to perform multi-digit arithmetic.

In the previous experiment you constructed a full-adder and burned it into a chip. If you use two such full-adders you can make an adder that will work with two two-digit numbers as shown below.



$$\begin{array}{r} X_1\ X_0 \\ +\ Y_1\ Y_0 \\ \hline C_2\ S_1\ S_0 \end{array}$$

Adder Circuit for Two Two-Digit Numbers

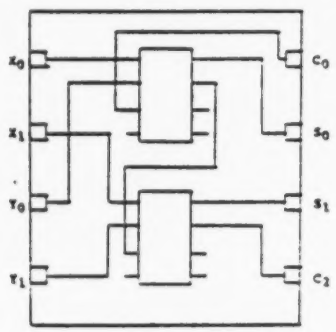
Notice that the carry output of the first full-adder is used as the input to the next one. This is known as "ripple carry". The output of the second full-adder will not be valid until the carry from the first has been generated and processed. Study this circuit until you understand it fully.

We need not stop at two digits. Additional full-adders can be added to extend the addition to any desired length. As we do this, the ripple carry delay becomes more and more significant.

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab.

Load the full-adder created in the last experiment into two chips (numbers 1 and 2). Take these two chips inside the Prototype Chip and wire them as shown below.



Two-Digit Full-Adder Construction

Go outside the Prototype Chip. Apply the appropriate logic levels (toolkit flip-flops are useful for this purpose) to the pins of the Prototype Chip to perform the following addition.

```
  0 1
+ 1 0
-----
```

Calculate the expected result and write it below.

Does this correspond with your measured result?

Now apply appropriate logic levels to the X and Y inputs for this addition:

```
  1 1
+ 1 1
------
```

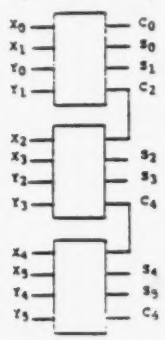
Calculate the expected result and compare it with your experimentally determined result.

Apply logic 1 to pins 2 and 3. Leave pins 1 and 4 unconnected (logic 0). What are the outputs on pins 5, 6, and 7?

Now apply the hot cursor to pin 1. Note that pin 5 goes high after a short delay. This delay is due to the ripple carry. Use a watch to measure the time it takes between application of the hot cursor and the appearance of logic 1 on pin 5. Enter the result below.

Burn the Prototype Chip into small Chip 3. Verify that the small chip behaves exactly like the prototype chip. Pick up the chip, press "?", and document it. Copy below what you have documented (or provide a printout).

Save the chip on your Storage Disk, then load it into chips 4 and 5. Connect chips 3, 4, and 5, as shown below.



Six-Bit Full-Adder Construction

Connect a logic 1 to the Y_0 - Y_5 inputs. Leave the X_0 - X_5 inputs unconnected. Compare the experimentally determined output with the calculated output.

Now apply the hot cursor to X_0. Note that C_6 goes high after a short delay. Use a watch to measure the delay, and enter the result below.

Explain this result based on your measurement for the two-bit full-adder.

How long do you think it would take for the final carry to be valid if you had constructed an 18-bit full-adder?

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 7

DECODERS AND DEMULTIPLEXERS

NAME_____ DATE_____

OBJECTIVES

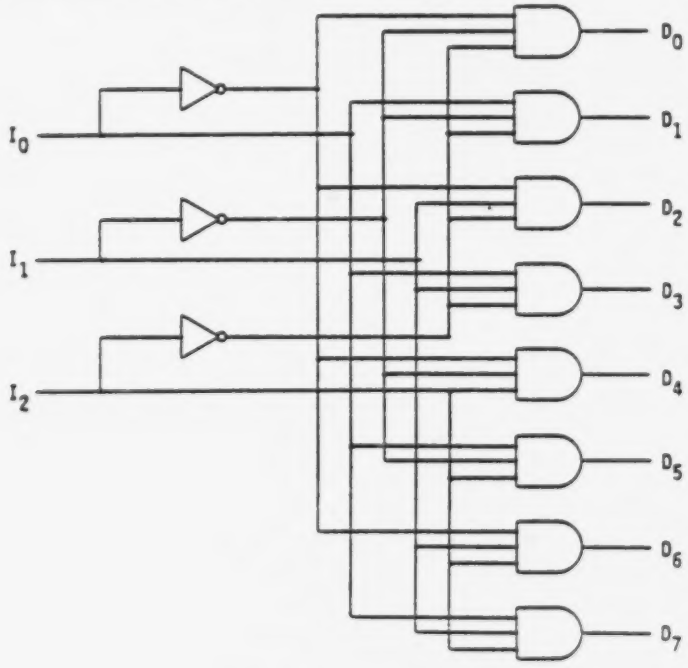
To study the properties and applications of some advanced combinational logic circuits.

PREPARATION

In digital computers (and elsewhere) data are frequently encoded in binary form. For example, the letter 'S' is coded as 1010011 in the American Standard Code for Information Interchange (ASCII). Decimal digits are frequently encoded in Binary Coded Decimal (BCD), with the decimal digits 0 through 9 being represented as 0000 through 1001 in binary.

It is sometimes desirable to convert the coded information back into discrete signals. A "decoder" circuit accomplishes this task. It converts binary information on n lines to a unique signal on one of 2^n output lines.

Shown below is a typical decoder.

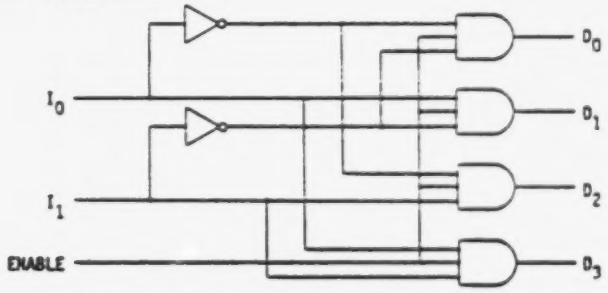


3-to-8 Line Decoder

For any combination of inputs on the three input lines $I_2 I_1 I_0$, exactly one of the eight output lines will be a logic 1. The D_0 line is logic 1 when the input is 000; the D_5 line is logic 1 when the input is 101. Spend a few minutes examining this circuit until you are sure you understand how it operates.

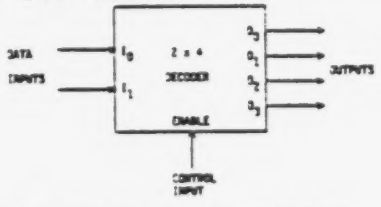
Since decoders perform such an important and frequently-used service, decoder circuits are implemented on commercial Medium Scale Integration (MSI) chips. This saves the digital designer the work and expense of building up the decoder from AND and NOT gates whenever he needs one.

Commercial decoder chips usually include an ENABLE input to control circuit operation. When the enable input is deactivated, it inhibits all outputs from the chip. The circuit below incorporates an ENABLE.

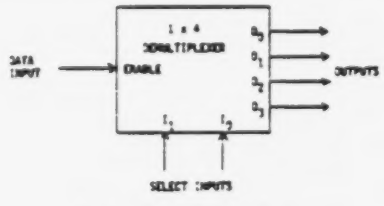


2-to-4 Line Decoder with ENABLE

The ENABLE input not only serves to allow the decoded output to be turned on and off, but it also allows the chip to perform a second function: demultiplexing. The figure below illustrates these two uses.



Decoder with ENABLE



Demultiplexer

The circuits are identical; the only difference is the way they are used. As a demultiplexer, the circuit receives information on a single line and transmits the information to one of the four outputs, depending on the Select Inputs. A circuit like this might be used in a telephone switching system where an incoming call could be routed to one of four phones.

Study the above circuits and descriptions until you clearly understand the functions of decoders and demultiplexers.

Commercial decoder/demultiplexers usually have inverted outputs. The "active" or selected output is a logic 0; all others are logic 1.

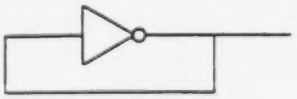
PROCEDURE

Boot Side 2 of the Robot Odyssey I disk and select the Innovation Lab. Go inside the Prototype Chip.

Construct a 2-to-4 decoder with ENABLE. Connect I_0 to pin 1, I_1 to pin 2, ENABLE to pin 4, and D_3 through D_0 to pins 5 through 8. Sketch (or provide a printout of) your circuit below. Include labels and written explanations where necessary.

Go outside the Prototype Chip and test the circuit by applying various logic levels to I_1 , I_0 , and ENABLE. Give the results of your test below, and reconcile them with the operations you expect from a decoder.

Connect the circuit below to the ENABLE pin.

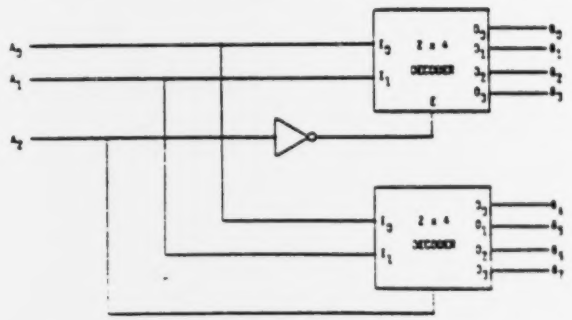


Apply various logic levels to I_1 and I_0. Discuss your results with reference to the behavior expected of a demultiplexer.

Burn the Prototype Chip into a small chip. Verify that the small chip functions correctly. Pick up the small chip, press "?", and enter your documentation. Copy below what you have documented (or provide a printout).

Save the chip on your storage disk. Make one more copy of the chip.

A 3-to-8 decoder can be made from two 2-to-4 decoders as shown below.



3-to-8 Decoder Construction

Construct this circuit and test its operation by applying various logic levels to $A_2A_1A_0$. Give the results of your test below, and explain how this circuit functions as a 3-to-8 decoder.

Decoders are occassionally useful in synthesizing Boolean functions. Add an OR gate to the 3-to-8 decoder circuit to synthesize the function $F = A_2A_1'A_0 + A_2'A_1A_0$. Sketch the circuit below (or provide a printout).

Test the circuit to determine how it functions. Describe your results below.

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 8

RS FLIP-FLOP

NAME_____ DATE_____

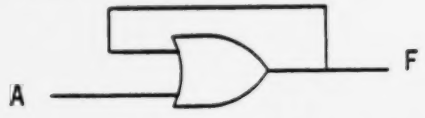
OBJECTIVES

To introduce sequential circuits.
To construct an RS flip-flop from gates.

PREPARATION

The experiments up till now have used only "combinational logic". In such circuits, the output depends only on the input. In "sequential" circuits, the output will depend not only upon the input, but also on the inputs that have occurred in the past. These circuits "remember" past events. They employ "feedback" to take a portion of the output signal and use it as an input.

Consider the circuit below.

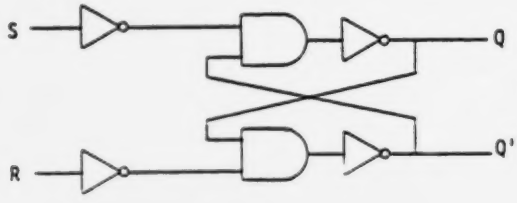


Logic Circuit with Feedback

The output of this circuit will depend on the history of the A input. If A has always been 0, then F will be 0. But if A ever was 1, even though it might return to 0, F would become 1 and then "latch" by feedback so that it would always remain 1. This circuit is not very useful, but it does illustrate the feedback concept in sequential circuits.

A flip-flop is a sequential circuit with two stable states, logic 0 and logic 1. Depending upon its current inputs and upon its previous inputs, the current output will either be a logic 1 or a logic 0.

The RS flip-flop can be constructed as shown below.



RS Flip-Flop Construction

To analyze this circuit, first assume that Q (output), S (set), and R (reset) are logic zeros and Q' is a logic one. The upper AND gate will then be fed by two 1's and will produce a 1 output. The upper right inverter will produce a 0 output for Q as we had assumed. The lower AND gate is fed by a 0 and a 1 producing a 0 output. The lower right inverter will thus produce a 1 output for Q' as required.

The circuit will remain in this state indefinitely. But now analyze what would happen if Q were originally a 1 and Q' a 0. Spend a minute or two to convince yourself that this is also a stable state for the flip-flop as long as R and S remain 0.

We thus conclude that when R and S are 0, the outputs Q and Q' will remain unchanged. We also notice that Q and Q' are the opposite of each other, that is why they have the same symbol, except for the prime.

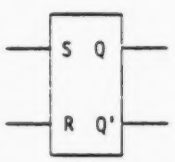
Let us once again assume R = 0, S = 0, Q = 0, Q' = 1. Now let us set S to 1. Notice that the upper AND gate will then be fed by 0, forcing its output to a 0, and forcing Q to 1. Now the lower AND gate will be fed by two 1's; its output will be 1, and Q' will become 0. We see that a 1 on the S (set) input, sets Q to 1.

With R = 1 and S = 0, we can follow similar logic to show that Q' will be forced to 1. R (reset) causes Q to be reset to 0.

The only other case to analyze is when both R and S are 1. This causes both Q and Q' to become 1, but if R and S simultaneously return to 0, the values of Q and Q' become difficult to predict. Therefore, for the RS flip-flop, the condition R = 1, S = 1 is usually considered to be "illegal" in the sense that a circuit designer will avoid this condition.

Below is shown the standard symbol for an RS flip-flop along with its truth table.

S	R	Q
0	0	No change
0	1	0
1	0	1
1	1	Illegal



RS Flip-Flop Truth Table and Circuit Symbol

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab.

Construct the RS flip-flop as shown in the Preparation section of this experiment. Be sure the remote control is on.

With Q initially 0, apply the hot cursor to the S input. What happens to Q?

Now remove the hot cursor. What happens to Q?

Again, apply the hot cursor to S. What happens?

Apply the hot cursor to R. What happens?

Remove the hot cursor from R. What happens?

Apply the hot cursor to S again, but this time, do so only for a brief instant. Describe the phenomenon that occurs. Analyze the circuit to determine why this happens.

You should now be aware of one additional property of flip-flops: "hold time". This is the time that the input must remain steady to assure that the flip-flop will operate correctly. For a real-world flip-flop, the hold time is typically 5 nanoseconds.

Now test the "illegal" input by applying the hot cursor to both R and S SIMULTANEOUSLY. Describe what happens.

Remove the hot cursor from R and S SIMULTANEOUSLY. What happens?

Take one of the flip-flops from the tool kit and compare it with the flip-flop you built. It is an RS flip-flop also, but its properties are somewhat different. Discuss the similarities and differences between these two flip-flops. Consider in particular if there are any hold time limitations on the tool kit flip-flop and how it behaves with an "illegal" input.

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 9

GATED AND CLOCKED FLIP-FLOPS

NAME_____ DATE_____

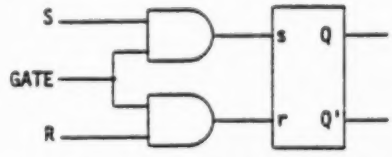
OBJECTIVES

To investigate the properties of gated and clocked flip-flops.

PREPARATION

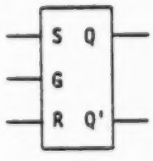
Frequently, flip-flops (and other chips as well) are "gated". The gate input controls the effect of the other inputs, such as R and S.

The circuit below shows how a basic RS flip-flop can be provided with a gate.



Gated RS Flip-Flop Construction

The inputs to the basic flip-flop will always be 0 as long as the gate input is a logic 0. The standard symbol for this kind of flip-flop is shown below.

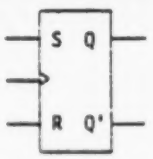


Gated RS Flip-Flop Circuit Symbol

When the gate is a logic 1, Q will be allowed to change under control of S and R. But, when the gate becomes a logic 0, Q will be "latched" at whatever its current value is; that is why this kind of flip-flop is sometimes called a latch.

The vast majority of flip-flops are "synchronous". They are rigidly timed and require a clock signal to function properly. The asynchronous flip-flops we have discussed so far allow changes in the output whenever the inputs change. Synchronous flip-flops change their states only at specific times controlled by the clock.

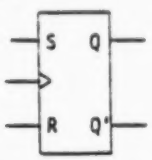
Although gated flip-flops are similar to clocked flip-flops, there is a fundamental difference. With a gated flip-flop, the output is controlled by the input as long as the gate is a logic 1, so many changes could occur in the output while the gate remains at 1. In contrast, clocked flip-flops allow but one change in the output per clock pulse regardless of how long the clock pulse may remain at logic 1. There are two basic techniques for accomplishing this: circuits may be edge-triggered or master/slave. Both have the same circuit symbol.



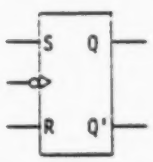
Edge-Triggered or Master/Slave RS Flip-Flop

The clock, which is usually a sequence of regularly spaced logic pulses, is connected to the triangle input.

Edge-triggered flip-flops respond not to a specific logic level at the clock input (as do gated flip-flops), but to a TRANSITION between logic levels. A flip-flop that responds to a transition from 0 to 1 is called a positive-edge-triggered flip-flop. If it responds to a 1 to 0 transition, it is called a negative-edge-triggered flip-flop. Their circuit symbols are shown below.



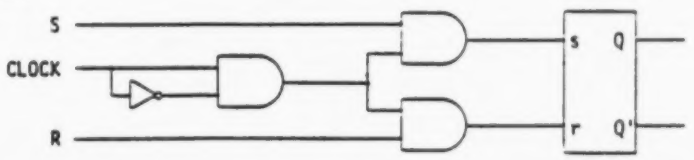
Positive-Edge-Triggered Flip-Flop



Negative-Edge-Triggered Flip-Flop

At the instant the transition occurs, the inputs are allowed to change the output. At all other times, the inputs are ignored and the output will remain unchanged.

We can simulate a positive-edge-triggered flip-flop with the circuit below.

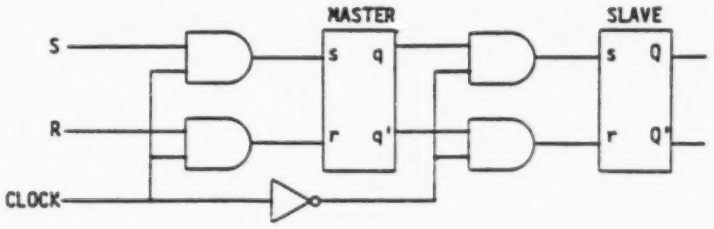


Positive-Edge-Triggered RS Flip-Flop Construction

While the clock is a logic 0, the inputs to the central AND gate are 0 and 1. When the clock pulse changes to logic 1, the inputs to the AND gate will be (briefly) 1 and 1; this is so because it will take a short but non-negligible amount of time for the

inverter to change its output from 1 to 0. The AND gate will therefore provide a brief logic 1 which will allow the R and S inputs to be sensed by the basic flip-flop. But this pulse will quickly disappear causing 0's to appear once again as inputs to the basic flip-flop. IMPORTANT: Read this paragraph again if you didn't fully understand it.

Another technique is used in master/slave flip-flops, which are sometimes called "pulse-triggered" flip-flops. The master/slave flip-flop can be constructed from two basic flip-flops, as shown below.



Master/Slave RS Flip-Flop Construction

While the clock is at logic 0, the output of the inverter is 1. This enables the master to control the slave, and the outputs are Q = q and Q' = q'. The inputs to the master flip-flop are disabled. When the clock becomes 1, the inputs to the slave are disabled, latching the output at its present value, while the inputs to the master are enabled. When the clock drops to 0 again, the new value latched into the master flip-flop once again appears as the output of the slave.

In normal operation, the flip-flop works this way: While the clock is low, new inputs to S and R are set up. When the clock goes high, the new inputs are latched into the master flip-flop. When the clock goes low, the master controls the slave, and the new output appears. IMPORTANT: Reread the above and study the circuit diagram until you thoroughly understand the master/slave flip-flop.

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab.

Using a flip-flop and two AND gates from the tool kit, construct the gated RS flip-flop shown in the Preparation section of this experiment. Complete the truth table below by applying the hot cursor for all possible cases of Gate, R, S, and initial value of Q.

Gate	S	R	Q
0	0	0	
0	0	1	
0	1	0	
0	1	1	Illegal
1	0	0	
1	0	1	
1	1	0	
1	1	1	Illegal

You should have noticed that the final value of Q depends (in certain cases) on the original value of Q. Make appropriate notes on your truth table to account for these cases if you have not already done so.

Explain (in words) the gated flip-flop's operation based on your measurements.

Rewire the gated flip-flop as a positive-edge-triggered RS flip-flop as shown in the Preparation section of this experiment.

Start with clock = 0, Q = 0. Now change S and R as desired. What happens to Q?

With S = 1, R = 0, Q = 0, change the clock from 0 to 1. What happens to Q?

Now change S and R as desired. What happens to Q?

Wire up the master/slave RS flip-flop given in the Preparation section of this report.

With the CLOCK input held at 1, describe the behavior of the Master portion of this master/slave flip-flop as you change R and S with the hot cursor.

With the clock set at 0, describe the behavior of the master portion of the flip-flop to changes in S and R.

Start with S = R = Q = CLOCK = 0. Now set S = 1, then set CLOCK = 1, then set CLOCK = 0. Describe (in words) the behavior of the flip-flop as you go through these settings.

Compare the behavior of this flip-flop to the behavior you would expect from the positive-edge-triggered flip-flop you constructed earlier (if subjected to the same sequence of inputs).

Once again start with S = R = Q = CLOCK = 0. Now set CLOCK = 1, then set S = 1, then set S = 0, then set CLOCK = 0. Describe the behavior of the flip-flop as you go through these settings.

Compare this behavior to what you would expect from your edge-triggered flip-flop (there should be a striking difference).

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 10

D AND JK FLIP-FLOPS

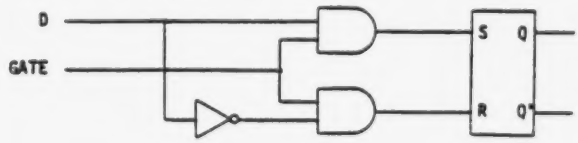
NAME_____ DATE_____

OBJECTIVES

To design, construct, and study the properties of D and JK flip-flops.

PREPARATION

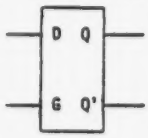
The D (for delay) flip-flop is available commercially as both an asynchronous (gated) or synchronous (usually edge-triggered) device. The gated version is often simply called a "latch" instead of a flip-flop. A gated D flip-flop circuit can be constructed as shown below.



Gated D Flip-Flop Construction

As long as the gate is at logic 0, D will be disabled. When the gate is at logic 1, the output Q will be the same as D.

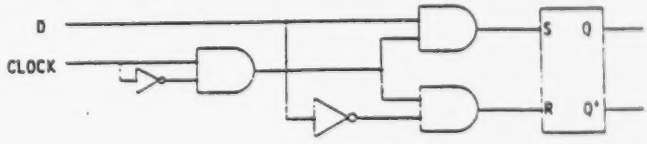
A gated D flip-flop has the circuit symbol shown below.



Gated D Flip-Flop Circuit Symbol

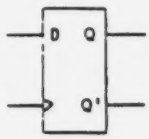
In a typical application, this device is used as a latch; Q follows D until the gate is brought low to latch in the current value of D.

Below is shown a positive-edge-triggered D flip-flop constructed from a simple RS flip-flop.



Positive-Edge-Triggered D Flip-Flop Construction

A D flip-flop such as this has the circuit symbol shown below.

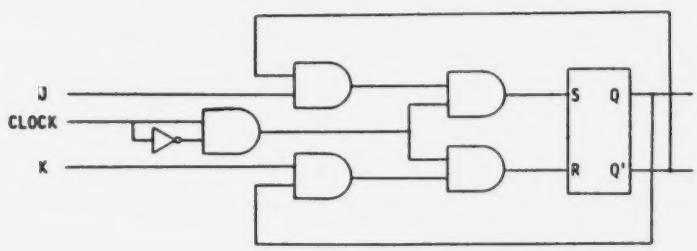


Positive-Edge-Triggered D Flip-Flop Circuit Symbol

In this circuit, Q will be set to the current value of D on a clock transition from logic 0 to logic 1. Changes in D will have no effect while the clock is held at a constant level. Only a positive clock edge can produce an effect on the output.

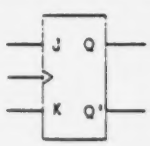
The JK flip-flop is arguably the most common of all flip-flops, owing to its great flexibility. It is a refinement of the RS flip-flop. The "illegal" state of the RS flip-flop (both inputs equal to one) has a specifically defined effect for the JK flip-flop: it causes the flip-flop to "toggle" (change states). In other respects, the JK flip-flop is exactly the same as the RS flip-flop.

The circuit below will simulate the behavior of a positive-edge-triggered JK flip-flop.



Positive-Edge-Triggered JK Flip-Flop Construction

It has the circuit symbol shown below.



Positive-Edge-Triggered JK Flip-Flop Circuit Symbol

Its truth table looks like this:

J	K	Q
0	0	Q (No change)
0	1	0
1	0	1
1	1	Q'(Change state)

Truth Table for JK Flip-Flop

Spend a few minutes going through the logic to convince yourself that this flip-flop will operate as indicated in the truth table. The Q in row 1 of the truth table indicates that when J = 0 and K = 0, the flip-flop remains in whatever state it is in. The Q' in row 4 indicates that when J = 1 and K = 1, the flip-flop will "toggle" (change state) after each positive clock edge.

Note that it is not possible to make a simple asynchronous JK flip-flop. If the clocking circuitry is removed, the flip-flop will function as expected for every case, except where J = 1 and K = 1. In that case, the flip-flop will continuously toggle back and forth between the 0 and 1 states. A gated version of the flip-flop will have the same problem.

Be sure you fully understand the operation of the D and JK flip-flops before going on.

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab.

Construct the gated D flip-flop as shown in the Preparation section of this experiment.

Set the gate to 0. What happens to Q when you apply the hot cursor to D?

Set the gate to 1. What happens to Q as you alternately apply and remove the hot cursor at the D input?

Modify the circuit to make it a positive-edge-triggered D flip-flop, as shown in the Preparation section of this experiment.

Set the clock input to 0. How does Q behave as you change D?

Set the clock input to 1 (ignoring any transition in Q due to this change in the clock). How does Q behave as you change D?

With Q = 0 and D = 1, change the clock input from 0 to 1. What happens?

With Q = 0 and D = 1, change the clock input from 1 to 0. What happens?

What is the fundamental difference between the gated and edge-triggered D flip-flop?

Go inside the Prototype Chip and wire the positive-edge-triggered JK flip-flop as shown in the Preparation section of this report. Be careful and neat in this construction; you will use it many times in later experiments.

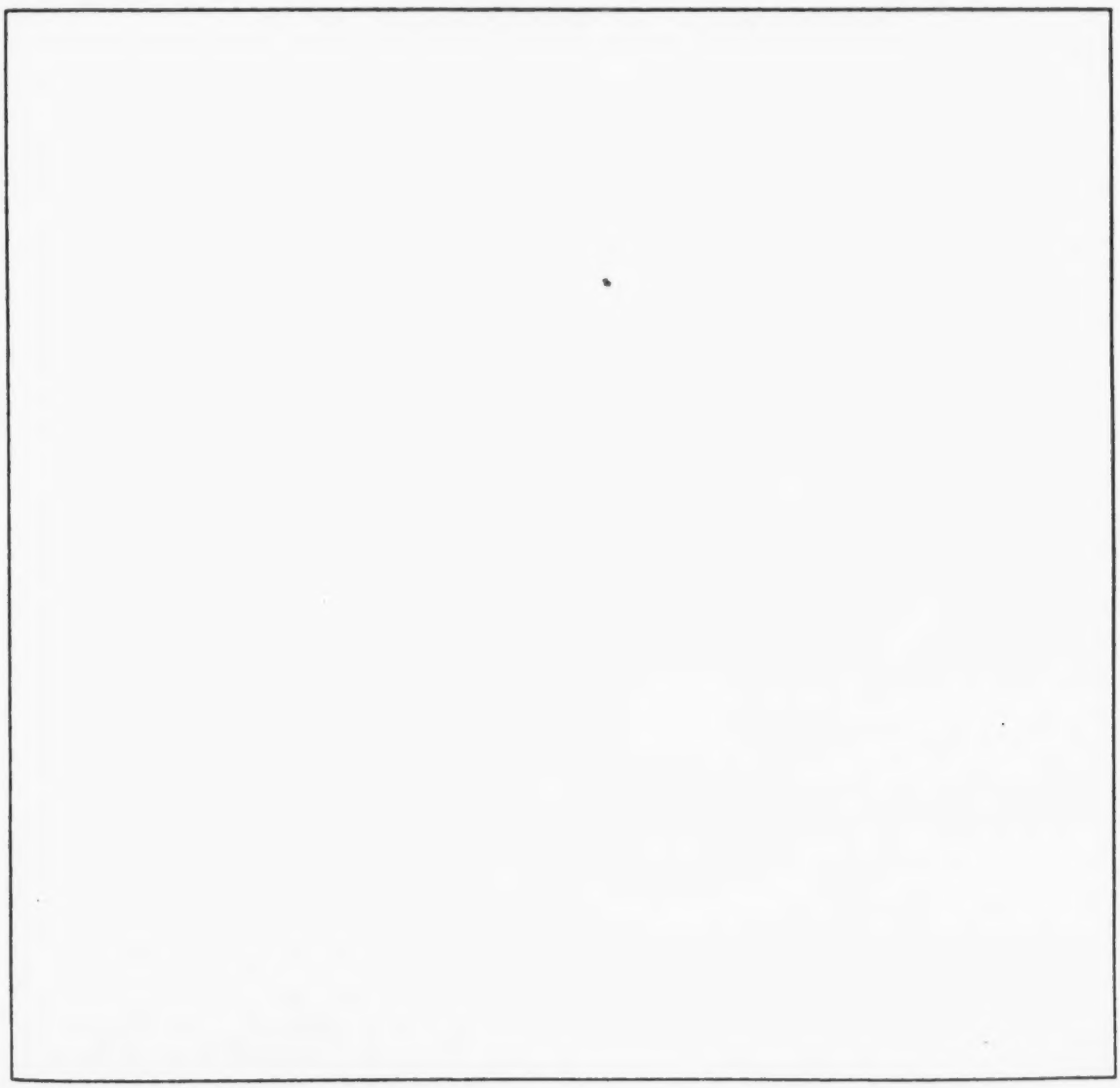
Use the hot cursor to test the circuit. Complete the truth table below based on this test.

J	K	Q
0	0	
0	1	
1	0	
1	1	

You should have noticed that for certain values of J and K, the new value of Q depends on the old value of Q. If you have not observed this, go back and make some more tests, then make appropriate corrections to your truth table.

Frequently, commercial JK flip-flops are equipped with asynchronous Preset and Clear inputs. The Preset input causes the flip-flop to be set to 1 regardless of the states of the inputs or the clock. The Clear input causes the flip-flop to be reset to 0 regardless of the inputs or the clock.

Design your own logic circuitry to add Preset and Clear inputs to this flip-flop. Test your circuit to make sure it functions as desired. Sketch the complete circuit below (or provide a printout). Be sure it is properly labeled.



Connect the 5 inputs and 2 outputs of the circuit as follows: J, K, PRESET, CLEAR, and CLOCK to pins 1 through 5, respectively; Q' and Q to pins 7 and 8, respectively. Burn the chip into a small chip. Test to be sure the small chip functions as it should. Document the chip. Copy your documentation below (or provide a printout).

Save the chip on your storage disk.

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 11

REGISTERS

NAME_____ DATE_____

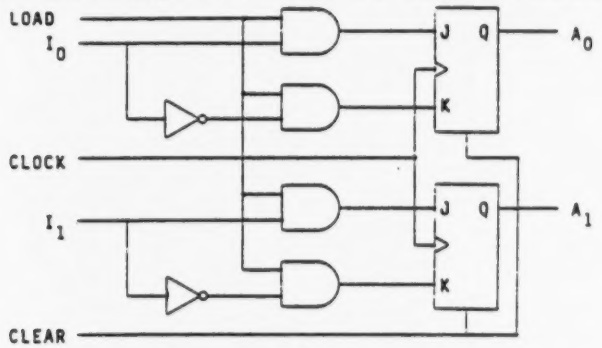
OBJECTIVES

To design registers similar to those used in computers.

PREPARATION

A register is a group of binary storage cells suitable for holding binary information. Registers are frequently used in the CPUs of computers to store information for various purposes. Typically, a CPU has an instruction register, a data register, an address register, and an accumulator which is a multipurpose register. The binary storage cells of a register are flip-flops. An n-bit register will have n flip-flops. The registers used in computers are usually synchronous, i.e., they are rigidly timed with clock signals.

The first register we will discuss is a synchronous 2-bit register with parallel load. Such a register might be suitable for temporary storage of data in a computer. The logic diagram below shows how this register can be implemented.

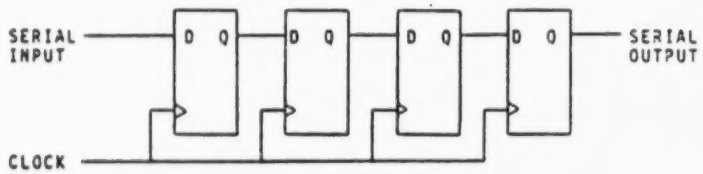


Synchronous 2-Bit Register

When the Load input is logic 1, I_0 and I_1 activate J or K as appropriate to set or reset the flip-flops. When Load is logic 0, the inputs are disabled, zeros are presented to J and K, and the flip-flops retain their current values. The clock is presumed to be a regularly occurring sequence of pulses.

A shift register is capable of shifting information either to the right or left (or sometimes both). It is made from a chain of flip-flops. Shift registers have many useful applications. They can be used for serial/parallel and parallel/serial conversions; they are used in binary multiplication and division.

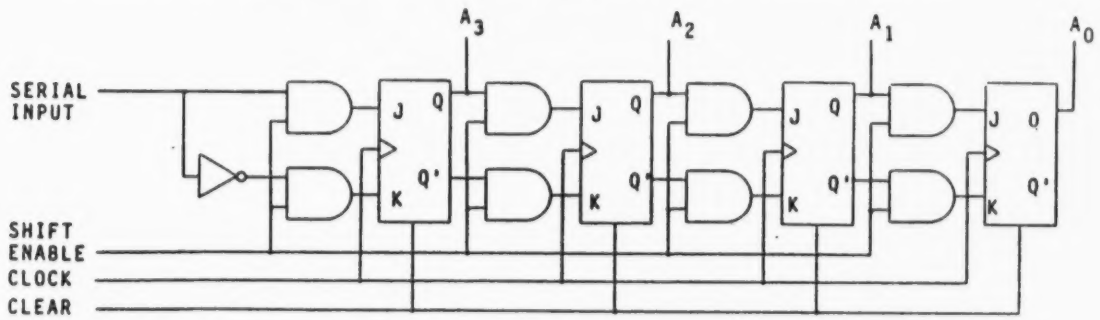
A simple shift register is shown below.



Simple Shift Register

With each clock pulse, the contents of each D flip-flop is transferred to the next one. If we want to control the shift, we could control the clock inputs by adding an AND gate to the clock line. This may be undesirable as it makes the shift register no longer a synchronous circuit in the strict sense.

A more sophisticated shift register is shown below.



Sophisticated Shift Register

This shift register has serial input and parallel output; it is synchronous (rigidly timed); it has an input to enable and disable a shift at each clock pulse. Examine this logic circuit and make sure you understand its operation.

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab. Go inside the Prototype Chip taking 2 copies of the JK flip-flop you constructed in an earlier experiment.

Construct the 2-bit register as shown in the Preparation section of this experiment.

Use the hot cursor to clear the register. Apply appropriate signals to CLOCK, LOAD, I_0, and I_1 to test the register.

Describe (in words) how the circuit behaves with LOAD = 0 as you vary the other parameters.

Describe (in words) how the circuit behaves with LOAD = 1 as you vary the other parameters.

Describe the effect of the CLOCK input.

Rewire the circuit to make a sophisticated shift register similar to the one shown in the Preparation section of this experiment, but use only two JK flip-flops to make a two-bit register.

Sketch your circuit (or provide a printout) below. Use labels and written explanations.

Clear the shift register. Connect a logic 1 to the serial input and to the SHIFT ENABLE input. Describe what happens as you apply pulses to the CLOCK input.

Now, set the serial input at 0 and repeat the above procedure.

Connect the 4 inputs and 2 outputs to appropriate pins of the Prototype Chip. Burn the Prototype Chip into a small chip. Test the small chip to be sure it functions properly. Document the chip. Copy your documentation below (or provide a printout).

Save the chip on your storage disk.

How many possible states are there for this register, i.e. how many possible combinations of ones and zeros are there?

Starting with the shift register cleared, connect an inverter between the A' output and the SERIAL INPUT. With SHIFT ENABLE = 1, apply clock pulses. List the states (values of A_0 and A_1) through which the shift register progresses. Describe (in words) how it works.

This configuration is called a Johnson counter or a switch-tail counter. If the shift register had 4 flip-flops, how many states would it have gone through before repeating?

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 12

COUNTERS

NAME_____ DATE_____

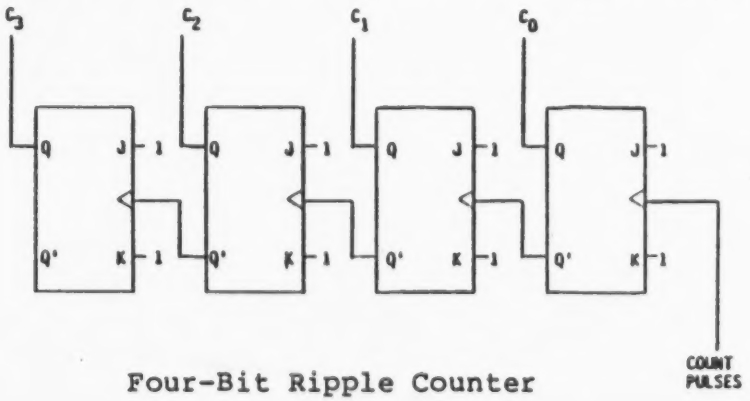
OBJECTIVES

To investigate the properties of asynchronous and synchronous counters.

PREPARATION

Counters of various sorts find frequent applications in logic design. For example, they are used to provide timing signals in computers and to provide synchronization for serial data transmission. Counters may be asynchronous or synchronous.

An asynchronous counter, also called a ripple counter, is shown below.



Four-Bit Ripple Counter

This circuit has 16 states and counts according to the sequence 0000, 0001, 0010, ..., 1111. It then starts over again from 0000. Note that J=K=1 for all flip-flops, so each flip-flop toggles (changes state) when it detects a positive clock-edge.

Starting from the initial 0000 state, an input count pulse will cause the rightmost flip-flop to toggle to the logic 1 state, driving Q high and Q' low. A negative clock-edge will appear at the clock input of the next flip-flop, so it will not toggle. The counter is now in the 0001 state.

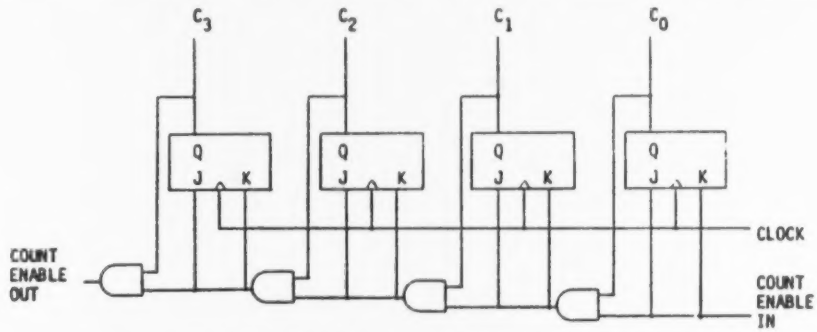
The next count pulse will again toggle the rightmost flip-flop. It goes to the logic 0 state forcing Q low and Q' high. The positive edge of Q' will cause the next flip-flop to toggle. The counter is now in the 0010 state.

Further count pulses will have similar effects. Note that the output from each successive flip-flop occurs slightly later in time than each previous flip-flop. That is why this is called a ripple counter.

This counter is asynchronous because the clock inputs of the flip-flops are used as part of the counter logic and because they are not all connected to a common clock.

Be sure you understand the operation of this counter before going on.

The circuit below is a 4-bit synchronous binary counter.



Four-Bit Synchronous Binary Counter

This circuit counts through 16 states, just as did the ripple counter. However, all four C outputs change at the same time and are rigidly controlled by a common clock.

If the COUNT ENABLE IN signal is a logic 1, each clock pulse will toggle the rightmost flip-flop. Each alternate clock pulse will toggle the next flip-flop, each fourth clock pulse will toggle the next, etc.

The COUNT ENABLE OUT signal is provided so that these counters can be cascaded, creating circuits that can count to 32, 48, 64, etc.

Go over the operation of this counter until you are sure you understand its operation.

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab. Make three copies of the JK flip-flop you created in a previous experiment.

Construct a 3-bit ripple counter similar to the ripple counter shown in the Preparation section of this experiment. You may find it convenient to use the Bus Chip stored on the Robot Odyssey disk to help provide the J and K inputs for the flip-flops.

Describe what happens as you apply pulses (the hot cursor) to the COUNT input. Note any "ripple" that occurs.

Apply count pulses until $C_2C_1C_0$ = 111. Now apply another count pulse. Describe what happens and measure (with a watch) how long it takes before all outputs are logic 0. Discuss your results.

Rewire the circuit to make a 3-bit synchronous counter similar to the synchronous counter shown in the Preparation section of this experiment. Include a CLEAR signal to allow all flip-flops to be cleared simultaneously.

Apply a logic 1 to COUNT ENABLE. Use the hot cursor to apply logic ones and zeros to the CLOCK input. Describe the results below, comparing the results to the ripple counter. Note any unusual occurrences.

In addition to the expected results for counting and activation of COUNT ENABLE OUT, you should also have observed that the COUNT ENABLE OUT signal flashes briefly at various times during the cycle. Give a detailed and well-reasoned explanation of why this happens. To do this you may need to follow the path of pulses in the circuit.

Use this circuit and appropriate logic gates to make a modulo-six counter. The circuit should count from 000 to 101, then start over. Each clock pulse should cause the count to increase by 1. Note: It is acceptable for the counter to briefly enter the 110 state before going to the 000 state, but it is possible (and better) to design a circuit that avoids the 110 state. Sketch (or provide a hardcopy of) your circuit below. Include appropriate labels and (if necessary) written explanations.

Test the circuit to be sure it counts from 0 to 5 and resets.

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 13

MEMORY

NAME_____ DATE_____

OBJECTIVES

To design and test a simple random access memory (RAM).

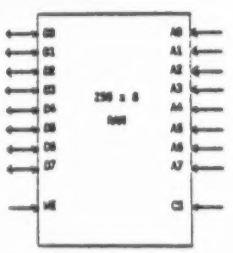
PREPARATION

Random access memory (RAM), more accurately called read/write memory, is one of the fundamental functional blocks of a computer. RAM provides storage for a computer's programs and data. Under control of the central processing unit (CPU) data can be stored in RAM (a write cycle) or retrieved from RAM (a read cycle). RAM performs no computational task; it functions only as storage.

Typical RAM memory chips are organized as "16k x 1" or "2k x 8", for example. Both of these chip organizations store 16384 bits of information (in computer jargon, k=1024). In the 16k x 1 chip, these data are stored as 1 bit at each of 16384 addresses, while in the 2k x 8 chip, the data are stored as 8 bits at each of 2048 addresses.

The purpose of the "addresses" of the RAM is so that specific data can be accessed by location. A typical microprocessor can access 64k (65536) memory addresses. This requires 16 address lines, because two to the 16th power is 65536. The 16k x 1 chip requires 14 address lines and the 2k x 8 chip requires 11.

A 256 x 8 RAM chip is shown below. It is quite similar to a commercially available RAM chip.



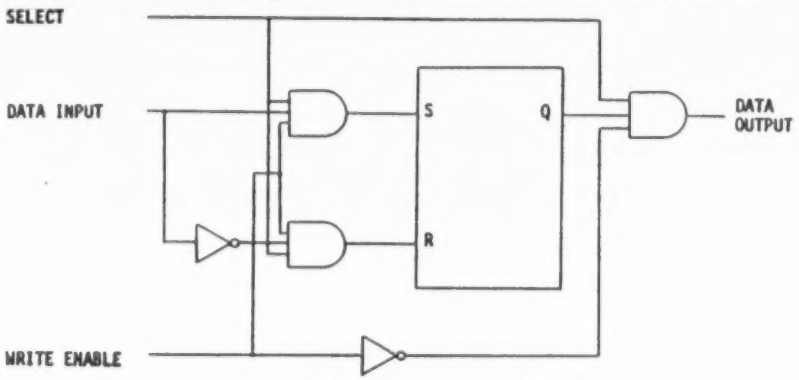
256 x 8 RAM Chip

This chip requires 8 address input lines to access its 256 locations. It requires 8 data lines to store or retrieve a "byte" of information. The WE (write enable) line tells the chip whether a read or a write operation is being performed. The CS

(chip select) line tells the RAM whether it is to be activated at all.

Note that the data lines are bidirectional; this is so that data can be stored or retrieved using the same set of lines. Due to limitations of the Robot Odyssey software, we cannot have bidirectional lines, therefore we must use separate lines for data input and output.

Each bit in a RAM chip, regardless of organization, is stored in a separate memory cell, such as the one shown below.

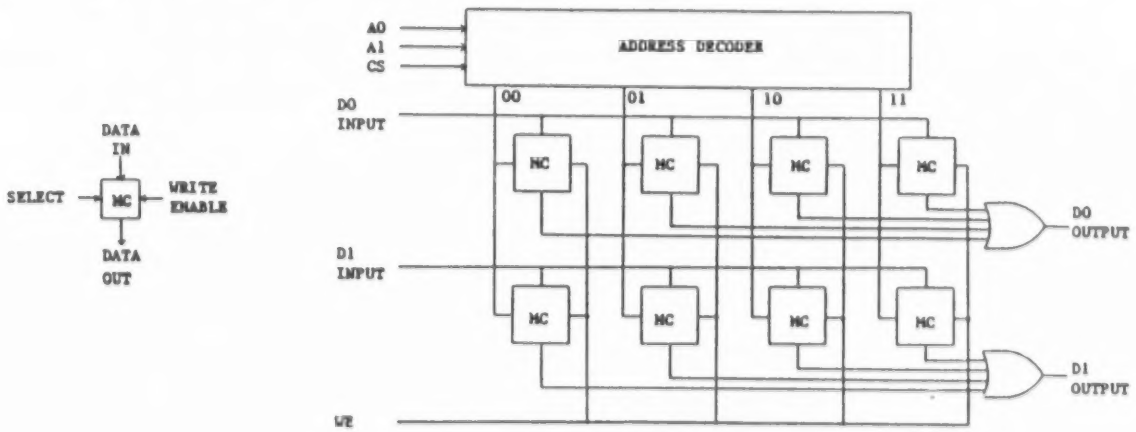


One-Bit Memory Cell

Data output will occur only when the cell is selected and "write enable" is off (a read cycle). In this case the bit stored in the flip-flop appears at the output.

The bit in the flip-flop can be changed if the cell is selected and if write enable is a logic 1. In this case a logic 1 is passed to the appropriate input (R or S) of the flip-flop according to the data input.

In a 4 x 2 memory (a very simple case) the memory cells would be connected as shown below.



4 x 2 Memory - Internal Organization

Carefully examine this circuit. The address decoder determines which pair of memory cells, if any, will be selected. No pair is selected unless the CS input is active. When writing to a pair of cells, the address decoder selects the appropriate pair while the D0 and D1 inputs supply the data. When reading, the OR gates supply the D0 and D1 outputs from the appropriate pair of cells.

PROCEDURE

Boot Side 2 of the Robot Odyssey I disk and select the Innovation Lab. Go inside the Prototype Chip.

Construct the memory cell from toolkit parts as shown in the Preparation section of this experiment. Because only two-input gates are available in the tool kit, certain modifications will be necessary.

Wire DATA INPUT to pin 1, SELECT to pin 4, DATA OUTPUT to pin 5, and WRITE ENABLE to pin 8. Go outside the Prototype chip. Describe below how you can load a logic 1 into this memory cell. Pay attention to the order in which the various signals must be applied.

Do what you have just described.

Describe below how you can read the contents of the memory.

Do what you have described. What is the observed output?

Burn the prototype chip into a small chip, test it to be sure it operates properly, and document the chip. Copy your documentation below (or provide a printout).

Save the chip on your storage disk.

Make 3 more copies of the memory cell chip. Wire a 2 x 2 memory similar to the one shown in the Preparation section of this experiment. It will be necessary for you to design your own address decoder. Make sure that both the output lines of the decoder are logic 0 when the CS input is logic 0.

Sketch (or attach a printout of) your logic diagram. Label your inputs as D0 IN, D1 IN, A0, CS, and WE. Label your outputs as D0 OUT and D1 OUT. Provide written explanations where necessary.

Describe how to load 01 into the memory at address 0 and then 10 into the memory at address 1. The order in which the signals are applied/removed is important.

Do what you have described.

Describe how to read memory at location 0 and then read memory at location 1.

Do what you have described. What is the observed output?

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.

EXPERIMENT 14

ROBOT CONTROL

NAME_____ DATE_____

OBJECTIVES

To apply the knowledge gained in the previous laboratories to a logic design problem.

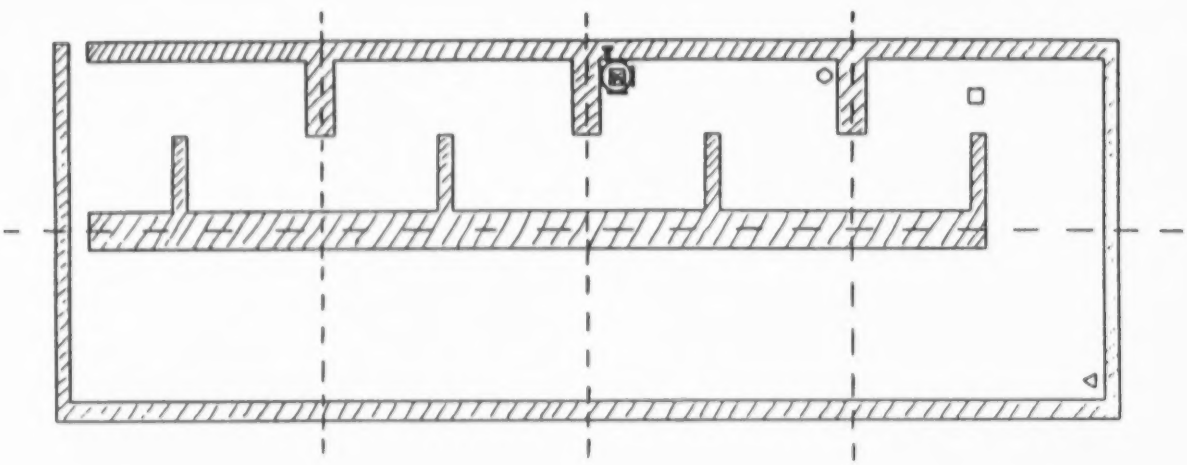
PREPARATION

The Robot Anatomy Tutorial was covered in the very first laboratory. At this time it may be a good idea for you to go back and review that tutorial, because you will now use your expertise with logic circuits to design a control system for a robot.

PROCEDURE

Boot side 2 of the Robot Odyssey I disk and select the Innovation Lab. Go to the Maze and take a robot with you.

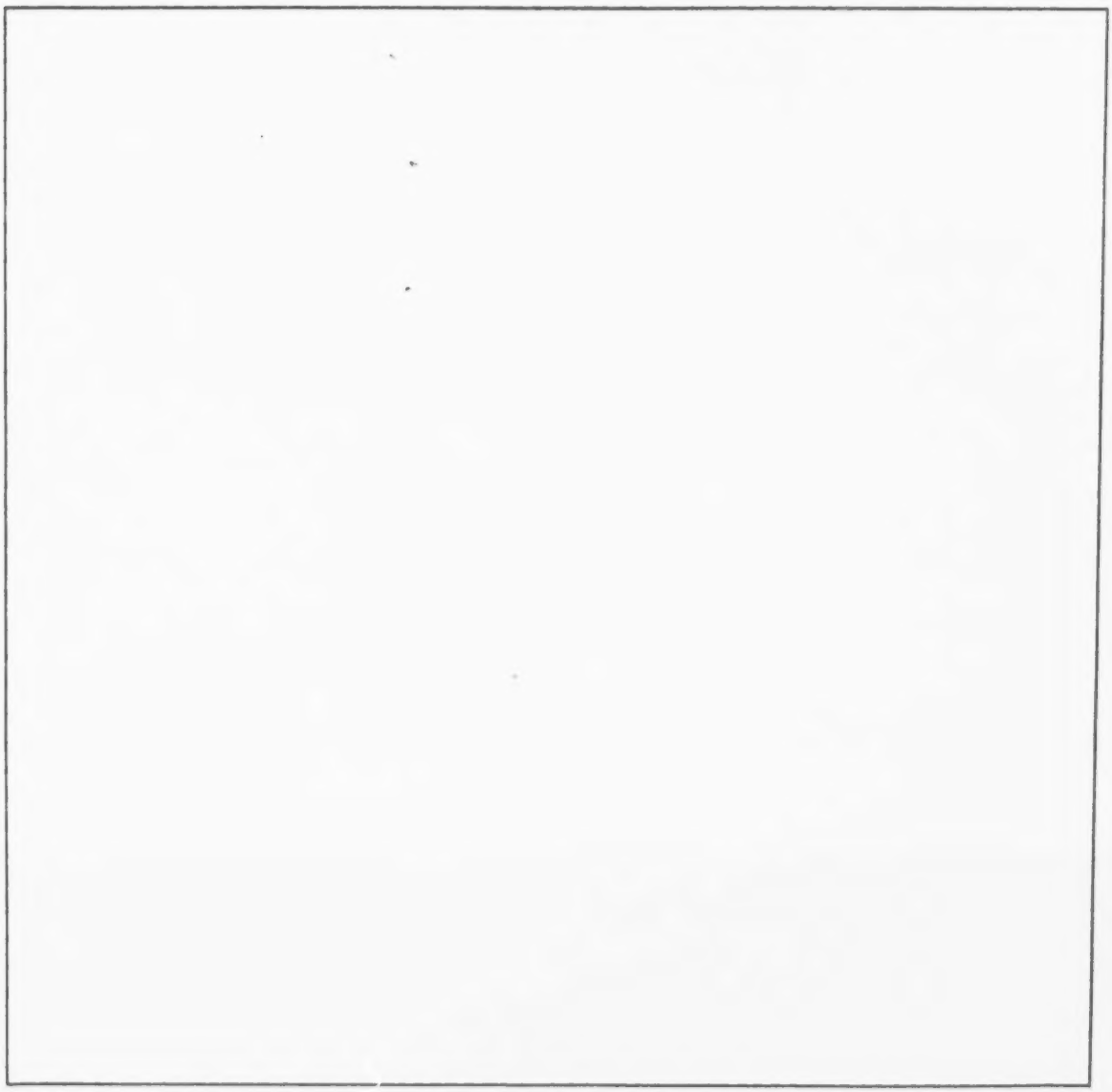
Place the robot, a triangle, a square, and a hexagon in the positions indicated on the Maze diagram below. The robot starts in the upper left corner with its bumpers touching both walls. The square is midway between the center wall and top wall.



MAZE

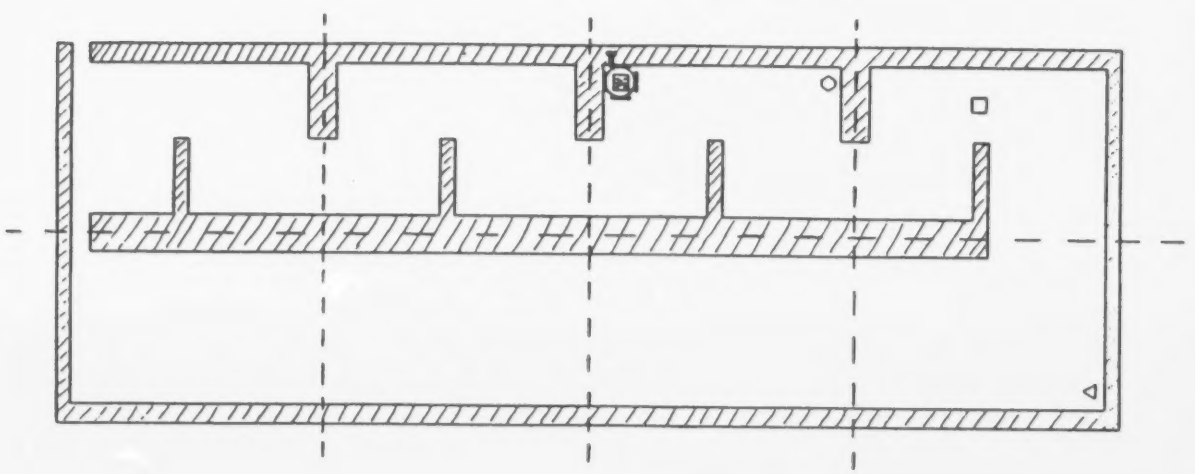
Design and install logic control circuitry inside the robot so that the robot will find and grab the square, then carry it to the room with the triangle, and drop it anywhere within that room. You may use toolkit parts, chips on the Robot Odyssey I disk, chips designed in previous experiments, and chips specially designed for this experiment. You may not exert any control over the robot once it has begun its task.

Sketch (or provide a printout of) the inside of the robot as wired for this task. Include appropriate labels and notes.



Describe how the control circuitry is supposed to work.

On the maze diagram below sketch the path taken by the robot.



MAZE

Use a watch to time the robot from start to the time the square is dropped. A good technique for this is to turn off the remote control, close the robot's power switch, move to the robot's eye, and then turn on the remote control. How much time did the trip take?

CRITIQUE

How long did it take you to do this experiment?

Please write a short paragraph describing your impressions of this experiment and giving suggestions as to how it could be improved.